Demonstration-Guided Q-Learning

Ikechukwu Uchendu[*†], **Ted Xiao**[†], **Yao Lu**[†], **Mengyuan Yan**[‡], **Joséphine Simon**[‡],
Matthew Bennice[‡], **Chuyuan Fu**[‡], **Karol Hausman**[†]
{ikechukwuu, tedxiao, karolhausman}@google.com

Abstract

In many challenging reinforcement learning (RL) settings, demonstrations are used to assist with exploration by allowing policies or value functions to directly learn from successful experience. In this work, we explore additional ways to utilize expert demonstrations to expedite training in value-based RL. In particular, we propose Demonstration-Guided Q-Learning (DGQL), an algorithm that noisily replays expert demonstrations to guide exploration and enable more efficient Q-value propagation in value-based RL methods. Contrary to common methods that utilize demonstrations in the context of value-based RL, we show that DGQL effectively leverages demonstrations to guide exploration via a replaying curriculum that relaxes common assumptions in simulated environments. In addition to analyzing the empirical benefits of more efficient value propagation, we show that DGQL is able to scale to difficult vision-based robotic manipulation tasks.

1 Introduction

While reinforcement learning (RL) has shown promising results in many domains, exploration is still one of the main limitations in applying RL to complex problems. A common approach to aid exploration is to use expert demonstrations to bootstrap the early stages of training [1]. Prior methods typically learn directly from demonstrations, such as via behavior cloning [2] or by mixing demonstrations directly with on-policy experience [3]. Are there other ways we can utilize information-rich expert demonstrations?

In this work, we propose that in addition to helping with exploration, expert demonstrations can also play a vital role in value propagation in value-based RL methods. Prior methods have considered resetting to exact states previously seen to create an exploration curriculum of increasing task difficulty. However, exact resets are often infeasible in dynamic domains such as robotic manipulation. Instead of using exact resets to form a curriculum of discrete start states, we *noisily replay* expert demonstrations to produce a curriculum of initial state distributions that not only provide exploration benefits, but also allow for more efficient dynamic programming by learning more accurate Q-function targets. Value functions trained on easier state distributions may act as promising targets for value functions being trained on more difficult state distributions.

We study the effect our method has on value propagation in a grid-world environment, demonstrating the empirical benefits of our method when paired with value-based RL algorithms that use dynamic programming. Then, we explore the performance of our method in a difficult vision-based simulated robotics environment. Our contributions are two-fold: (1) we introduce a new way of utilizing demonstrations in value-based RL settings and compare it to competitive value-based IL-RL methods, (2) we relax the simulation-specific assumptions of the previously proposed exploration-based methods.

[*]Google AI Resident
[†]Google Brain, United States
[‡]X, Mountain View, California, United States

NeurIPS 2021 Workshop on Robot Learning: Self-Supervised and Lifelong Learning, Virtual, Virtual

2 Related Work

Imitation Learning Combined with Reinforcement Learning (IL+RL). Earlier works on combining imitation learning and reinforcement learning use expert demonstrations for initialization [1, 4], use different losses for demonstrations versus RL episodes [5], or apply offline value-based RL to a joint dataset of demonstrations and RL episodes [3]. More recent approaches [6, 7, 8] also incorporate offline suboptimal data and regularize the learned policy to stay close to the demonstration data distribution. We leverage demonstrations by replaying them noisily and partially, which acts as a curriculum for RL and enables more efficient Q-value propagation.

Curriculum Learning and Exact State Resets for RL. The ability to exactly reset to arbitrary states in simulation has helped with exploration in RL. Some works uniformly sample states from demonstrations as start states [9, 10, 5], while others generate curriculums of start states. This includes methods that start at the goal state and iteratively expand the start state distribution, assuming reversible dynamics [11, 12] or access to an approximate dynamics model [13]. Other works generate the curriculum from demonstration states [14]. Instead of resetting exactly to specific states, our method generates a curriculum by partially replaying demonstrations with intrinsic and/or added noise. This broadens the distribution of start states compared to exact resetting along a narrow set of demonstrations, making the learning process more robust. Our approach could be extended to the real world, where instantaneously resetting to an arbitrary state in the environment is impossible.

3 Demonstration-Guided Q-Learning

3.1 Preliminaries and Motivation

We use the standard Markov decision process (MDP) formulation, $M = (S, A, T, R, \gamma)$ [15]. In addition, we consider an initial state distribution, $P(s_0)$, where $s_0 \in S$.

One category of RL algorithms is Q-learning [16], which solves for the optimal Q-function,

$$Q^*(s, a; \theta) = \mathbb{E}\left[R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a'; \theta)\right], \tag{1}$$

where θ parameterizes the Q function (e.g., a neural network). The optimal policy, π^*, is then given by $\pi^*(s) = \text{argmax}_{a \in A} Q^*(s, a; \theta)$.

In Q-learning, it is common for the agent to start in a state sampled from $P(s_0)$ and explore the environment with an epsilon-greedy strategy. In environments with sparse rewards, this approach has two shortcomings: (1) the Q-network starts learning only after acquiring successful episodes by chance; (2) learning relies on value propagation backwards from goal states and uniformly sampling transitions from all experiences is inefficient [17].

Adding a reverse curriculum to RL helps with both issues. It reduces the time to acquire initial successes by reducing the expected task horizon, and this shortened horizon simplifies value propagation since transitions with reward are more likely to be sampled. In the following curriculum stages, the Q-value targets will be computed using previously visited states, which helps propagate value to the newly expanded state region.

3.2 Method

Given demonstrations $\{\tau_1, \tau_2, \ldots, \tau_N\}$, we choose an integer K that denotes the number of curriculum stages used during training. While training, instead of sampling from the initial state distribution, we *noisily replay* a trajectory τ_i until step $T_{i_L} = |\tau_i| \times \frac{L}{K}$, where $|\tau_i|$ is the number of timesteps in trajectory τ_i, and $L = K - 1$. T_{i_L} is then the timestep that trajectory τ_i is replayed to when the agent is on curriculum stage L. We start with curriculum stage $K - 1$ since it gives the agent a chance to learn a region around the goal state, and a full noisy replay could put the agent in an undesirable position. After the replay is done, the policy starts exploring from the state $s_{T_{i_L}}$ reached by the noisy replay. Once the agent can reliably reach the goal after replaying to step T_{i_L} for all $i \leq N$, we replay until step $|\tau_i| \times \frac{L-1}{K}$. In practice, we move to the next curriculum stage when the average success for the current stage is greater than a threshold β. This process continues until we are starting from the initial state of τ_i which is in $P(s_0)$. We show pseudo code for our method in Algorithm 1.

3.2.1 Noisy Replays

Given a state, s, and action, a, NOISYREPLAY(s, a) executes action $a + \epsilon$ in the environment and returns the observed resulting state. s does not affect NOISYREPLAY and is only included to clarify the agent's current state. The noise, ϵ, can be inherent to the environment (e.g., from a stochastic simulator) or explicitly added to the actions (e.g., adding Gaussian noise). Given an expert demonstration $\tau_i = \{(\tilde{s}_1, \tilde{a}_1, \tilde{s}_2), \ldots, (\tilde{s}_{|\tau_i|}, \tilde{a}_{|\tau_i|}, \tilde{s}_{|\tau_i|+1})\}$, we execute NOISYREPLAY$(\cdot, \tilde{a}_i)$ for all $i \leq T_{i_L}$ at the start of each training episode corresponding with lines 8-11 of Algorithm 1.

Algorithm 1 Demonstration-Guided Q-learning

1: **Input:** expert demonstrations $\{\tau_1, \tau_2, \ldots, \tau_N\}$, threshold β, number of curriculum stages K, replay buffer D
2: Initialize value function $Q(s, a; \theta)$
3: $L \leftarrow K - 1$
4: **for** episode $= 1, M$ **do**
5: Randomly select expert demonstration $\tau_i = \{(\tilde{s}_1, \tilde{a}_1, \tilde{s}_2), \ldots, (\tilde{s}_{|\tau_i|}, \tilde{a}_{|\tau_i|}, \tilde{s}_{|\tau_i|+1})\}$
6: $T_{i_L} \leftarrow |\tau_i| \times \frac{L}{K}$
7: $s_{replay} \leftarrow \tilde{s}_1$
8: **for** $t = 1, T_{i_L}$ **do**
9: $\tilde{s} =$ NOISYREPLAY$(s_{replay}, \tilde{a}_t)$
10: $s_{replay} \leftarrow \tilde{s}$
11: **end for**
12: $s_t = s_{replay}$
13: **for** $t = 1, T$ **do**
14: Take action a_t using $Q(s, a; \theta)$ or a random policy with probability ϵ
15: Receive new reward r and state s_{t+1}
16: Append (s_t, a_t, r, s_{t+1}) to replay buffer D
17: **end for**
18: TRAINPOLICY(Q, θ, D)
19: $p =$ EVALUATEPOLICY$(Q, L, \{\tau_1, \tau_2, \ldots, \tau_N\})$
20: **if** $p >= \beta$ and $L > 0$ **then**
21: $L = L - 1$
22: **end if**
23: **end for**

4 Experiments

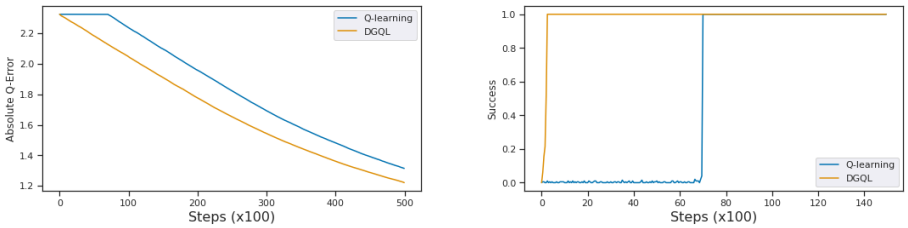
In our experimental evaluation, we study the following questions: (1) Does DGQL enable more efficient value propagation? (2) Does DGQL scale to complex vision-based robotic manipulation tasks? (3) How does DGQL compare to competitive IL+RL baselines?

(1) Value Propagation. In order to analyze how DGQL affects value propagation, we study simple grid-world environments where $V^*(s)$, the optimal value function, is tractable to compute. We train agents with DGQL and standard Q-learning; Figure 1 shows that DGQL learns a more accurate value function that enables it to solve the task. **(2) DGQL as Exploration in a Difficult Environment.** We study DGQL in a difficult simulated vision-based robotic grasping setting, described in A.1.2. In Figure 2(a), DGQL scales as an on-policy guided exploration method, while Qt-Opt cannot.

(3) Comparisons to IL+RL Baselines. Algorithms such as BC, AW-Opt, and CQL (see Sec.2) are competitive methods to apply in online settings when utilizing demonstrations with RL. In Figure 2(b), we see that DGQL is able to solve the task by using 100 expert demonstrations to guide exploration, while the IL+RL baselines are not able to solve the task with access to the same demonstrations.

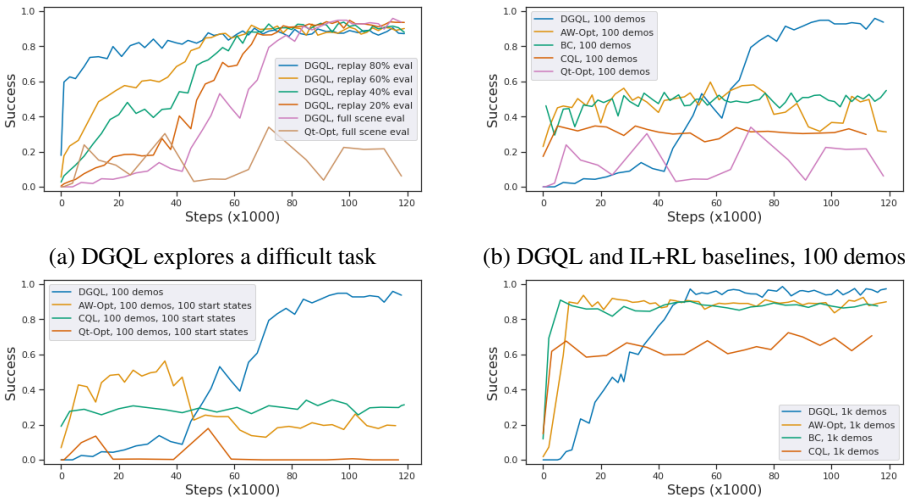
One possible explanation for this is that the baseline methods are trained and evaluated on a more complex distribution of randomly generated starting states, whereas DGQL is limited to 100 environment configurations due to the replay mechanism. In Figure 2(c), we limit the IL+RL baselines to initialize from one of the 100 demonstration scenes, but the baselines still do not match DGQL.

Since these IL+RL baselines allow policies or value functions to directly learn from demonstrations, they naturally scale well with more demonstrations. After providing 10 times more demonstrations (1000 total), the IL+RL baselines improve and nearly match DGQL, as seen in Figure 2(d).



(a) (b)

Figure 1: In toy grid-world environments, we study how DGQL affects value propagation and policy performance. (a) DGQL is efficient at learning accurate Q-values, measured with absolute Q-error, which compares learned Q-values with the optimal Q-values. (b) These more accurate Q-values allow DGQL to solve the task faster than standard Q-learning.



(a) DGQL explores a difficult task

(b) DGQL and IL+RL baselines, 100 demos

(c) Simplified starting state distributions.

(d) DGQL and IL+RL baselines, 1000 demos.

Figure 2: DGQL in a simulated robotic grasping environment. (a) DGQL uses demonstrations to guide exploration, as seen by the policy progressively performing better at intermediate replay evaluations and eventually solving the task, while Qt-Opt never does. (b) With access to 100 demonstrations, only DGQL is able to solve the task. (c) Limiting the set of initial states does not help the IL+RL baselines perform better. (d) With access to 10 times more demonstrations, the IL+RL baselines improve and nearly match DGQL.

5 Conclusion

We proposed Demonstration-Guided Q-learning (DGQL), a method for leveraging expert demonstrations to bolster exploration in value-based reinforcement learning. By creating a curriculum via *noisily replaying* demonstrations, we learned more accurate target values that enabled more efficient value propagation. We showed the benefits of DGQL in a robotic grasping task with a modest number of demonstrations, a common IL+RL setting. However, our method is not without limitations. We assume the ability to reset to initial states from expert demonstrations, which may not hold in transient environments. Furthermore, we assume that further replayed demonstrations are closer to goals, which may not hold for suboptimal demonstrations. Nonetheless, our assumptions are valid in many robotic situations, and our results suggest that demonstration-guided exploration may be a promising direction for future IL+RL work. We discuss future work for DGQL in Appendix A.2.

References

[1] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997.

[2] Nathan Ratliff, J Andrew Bagnell, and Siddhartha S Srinivasa. Imitation learning for locomotion and manipulation. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 392–397. IEEE, 2007.

[3] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards, 2018.

[4] Jens Kober, Betty Mohler, and Jan Peters. Imitation and reinforcement learning for motor primitives with perceptual coupling. In *From motor learning to interaction learning in robots*, pages 209–225. Springer, 2010.

[5] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[6] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[7] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. 2020.

[8] Yao Lu, Karol Hausman, Yevgen Chebotar, Mengyuan Yan, Eric Jang, Alexander Herzog, Ted Xiao, Alex Irpan, Mohi Khansari, Dmitry Kalashnikov, and Sergey Levine. Aw-opt: Learning robotic skills with imitation andreinforcement at scale. In *2021 Conference on Robot Learning (CoRL)*, 2021.

[9] Ionel-Alexandru Hosu and Traian Rebedea. Playing atari games with deep reinforcement learning and human checkpoint replay. *arXiv preprint arXiv:1607.05077*, 2016.

[10] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4), July 2018.

[11] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.

[12] Stephen McAleer, Forest Agostinelli, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube without human knowledge. 2019.

[13] Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.

[14] Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay:" man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

[15] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[16] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[17] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *arXiv preprint arXiv:2003.07305*, 2020.

Figure 3: Examples of grid-worlds used to analyze the effect that DGQL has on value function propagation. The grid on the left we call "line grid". The agent is able to move left, right, up, or down at each timestep. The goal state is labelled with a star.

A Appendix

A.1 Experiment Implementation Details

A.1.1 Grid-world

We use a standard grid-world setup where the agent is able to take one of four actions: left, up, right, and down. If the agent tries to move in the direction of a wall, nothing happens. For DGQL, we used a single demonstration and set $\beta = 0.8$ while letting K be equal to one more than the shortest path to reach the goal.

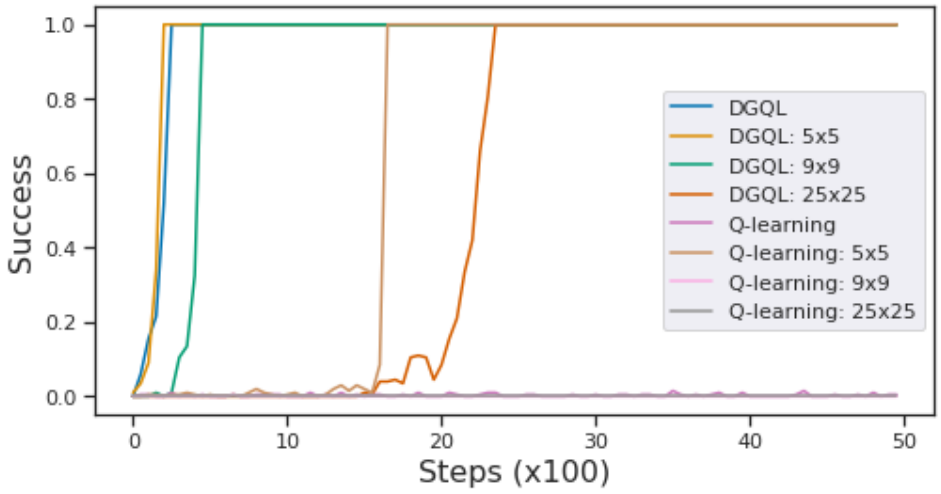
A.1.2 Simulated Robotic Manipulation

We simulate a 7 DoF arm with an over-the-shoulder camera (see Figure 4). Three bins in front of the robot are filled with various simulated objects to be picked up by the robot and a sparse binary reward is assigned if any object is lifted above a bin at the end of an episode. States are represented in the form of RGB images and actions are continuous Cartesian displacements of the gripper's 3D positions and yaw. In addition, the policy commands discrete gripper open and close actions and may terminate an episode.

A.2 Future Work

DGQL can be thought of as the combination of two policies: a guide-policy and an exploration-policy. The guide-policy moves the agent closer to the goal state, and the exploration-policy explores from where the guide-policy ends. Since the only criteria for the guide-policy is moving closer to the goal state, any policy superior to random exploration could be used as a guide-policy.

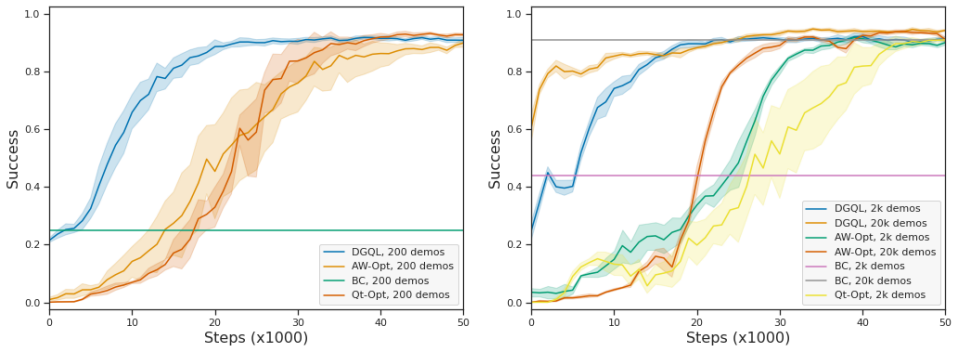
Behavioral cloning (BC) is a natural guide-policy that leverages expert demonstrations. To use BC as a guide-policy, we train a BC policy then run it in the environment to determine the average number of steps it takes for it to reach the goal. This value becomes the initial number of timesteps we roll out the BC policy for before switching to the exploration-policy. As the exploration-policy improves, we reduce the number of timesteps that the BC policy is rolled out for. Using BC as a guide-policy avoids the need to reset the environment to initial states from demonstrations. Figure 5 shows experimental results in the robotic grasping environment for DGQL using BC as a guide-policy as opposed to noisy replays.



(a)

(b)

Figure 4: (a): Grid-world results for DGQL on different environment difficulties comparing DGQL and standard Q-learning. (b): Simulated vision-based robotic grasping task with a 7DoF arm.



(a) Robotic grasping task, 200 demos

(b) Robotic grasping task, more demos

Figure 5: DGQL in the simulated robotic grasping environment but using Behavioral Cloning as a guide-policy as oppposed to noisy replays. (a) DGQL uses demonstrations to guide initial exploration, resulting in better sample efficiency even when only utilizing 200 demonstrations. (b) Utilizing 10 times and 100 times more demonstrations increases the performance of all offline-online methods, but DGQL is especially improved due to a stronger BC policy used to guide initial exploration.